LILO
Generic boot loader for Linux
Version 20

Technical overview

Werner Almesberger
werner.almesberger@lrc.di.epfl.ch

June 20, 1996

Contents

This document describes internals of LILO and related parts of its environment (kernel, etc.). It is not necessary to read or understand this document in order to install or use LILO. A general introduction and installation instructions can be found in the user's guide.

This document has only been partially updated and does not entirely reflect the current ('97) status of Linux or of LILO (version 20).

1 Load sequence

The boot sector is loaded by the ROM-BIOS at address 0x07C00. It moves itself to address 0x9A000, sets up the stack (growing downwards from 0x9B000 to 0x9A200), loads the secondary boot loader at address 0x9B000 and transfers control to it. It displays an "L" after moving itself and an "I" before starting the secondary boot loader. If a read error occurs when loading the secondary boot loader, a two-digit hex code is displayed after the "L". This results in an endless stream of error codes if the problem is permanent. Displaying these error codes is disabled if the build-time option `NO1STDIAG` is set.

The secondary boot loader loads the descriptor table at 0x9D200 and the sector containing the default command line at 0x9D600. If the default command line is enabled, its magic number is invalidated and the sector is written back to disk. This potentially dangerous operation can be disabled by defining `LCF_READONLY` when passing `second.S` through `cpp`. Next, the secondary boot loader checks for user input. If either the default is used or if the user has specified an alternate image, the options sector is loaded at 0x9D600 and the parameter line is constructed at 0x9D800. If the resulting line contains the option `lock`, the command line as entered by the user (it is saved before the final line is constructed) is written to the disk as the new default command line. Also, if a fallback command line is set, it is copied to the default command line sector.

If the user has supplied an initial RAM disk image, this file is loaded below the end of physical memory or 16 MB, whichever is lower. The start address is lowered to the next page boundary so that the memory area occupied by the initial RAM disk can later be easily returned to the system's free memory pool. The 16 MB limit exists because the BIOS functions used to transfer data in memory are only specified for an 24 bit address space.

Next, the floppy boot sector of that image is loaded at 0x90000[1], the setup part is loaded at 0x90200 and the kernel part is loaded at 0x10000, or, if the kernel has been compiled for being loaded "high" (i.e. with `make bzImage`), it is loaded at 0x100000 instead. During the load operations, the sectors of the map file are loaded at 0x9D000.

If the loaded image is a kernel image, control is transferred to its setup code. If a different operating system is booted, things are a bit more difficult: the chain loader is loaded at 0x90200 and the boot sector of the other OS is loaded at 0x90400. The chain loader moves the partition table (loaded at 0x903BE as part of the chain loader) to 0x00600 and the boot sector to 0x07C00. After that, it passes control to the boot sector.

Chain loaders that allow booting from a second drive (either floppy or hard disk) also install a small function to intercept BIOS calls and to swap the drive numbers at the top of available memory.

The secondary boot loader displays an "L" after being started and an "O" after loading the descriptor table and the default command line. Before loading the descriptor table, it checks, whether it has been loaded at the correct location and displays a question mark if it hasn't. If the descriptor table has an incorrect checksum, a minus sign is displayed.

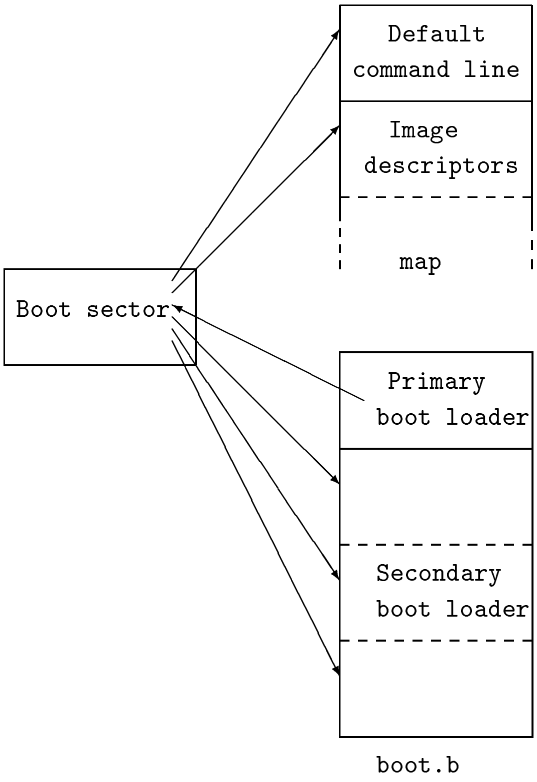
`0x00000`		1982 bytes
`0x007BE`	Partition table	64 bytes
`0x007FE`		29 kB
`0x07C00`	Boot load area	512 bytes
`0x07E00`		32.5 kB
`0x10000`	Kernel	448 kB
`0x90000`	Floppy boot sector	512 bytes
`0x90200`	Setup (kernel)	39.5 kB (2 kB used)
`0x9A000`	Primary boot loader	512 bytes
`0x9A200`	Stack	3.5 kB
`0x9B000`	Secondary boot loader	8 kB (3.5 kB used)
`0x9D000`	Map load area	512 bytes
`0x9D200`	Descriptor table	1 kB
`0x9D600`	Default command line, etc.	512 bytes
`0x9D800`	Keyboard translation table	512 bytes
`0x9DA00`	Parameter line construction area	1 kB
`0x9DC00`		7.5 kB
	Drive swapper	1 kB
`0xA0000`		

The area 0x90020-0x90023 is overlaid by a command-line descriptor while the secondary boot loader is running.

[1]The floppy boot sector is only used as a source of setup information.

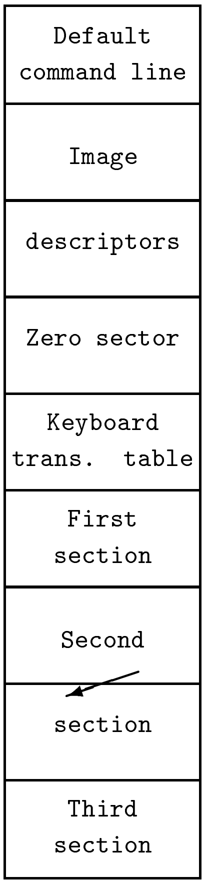
2 File references

This section describes the references among files involved in the boot procedures.



The boot sector contains the primary boot loader, the address of the default command line sector, the address of both descriptor table sectors and the addresses of the sectors of the secondary boot loader. The generic boot sector is copied from `boot.b`.

The primary boot loader can store up to eight sector addresses of the secondary boot loader.

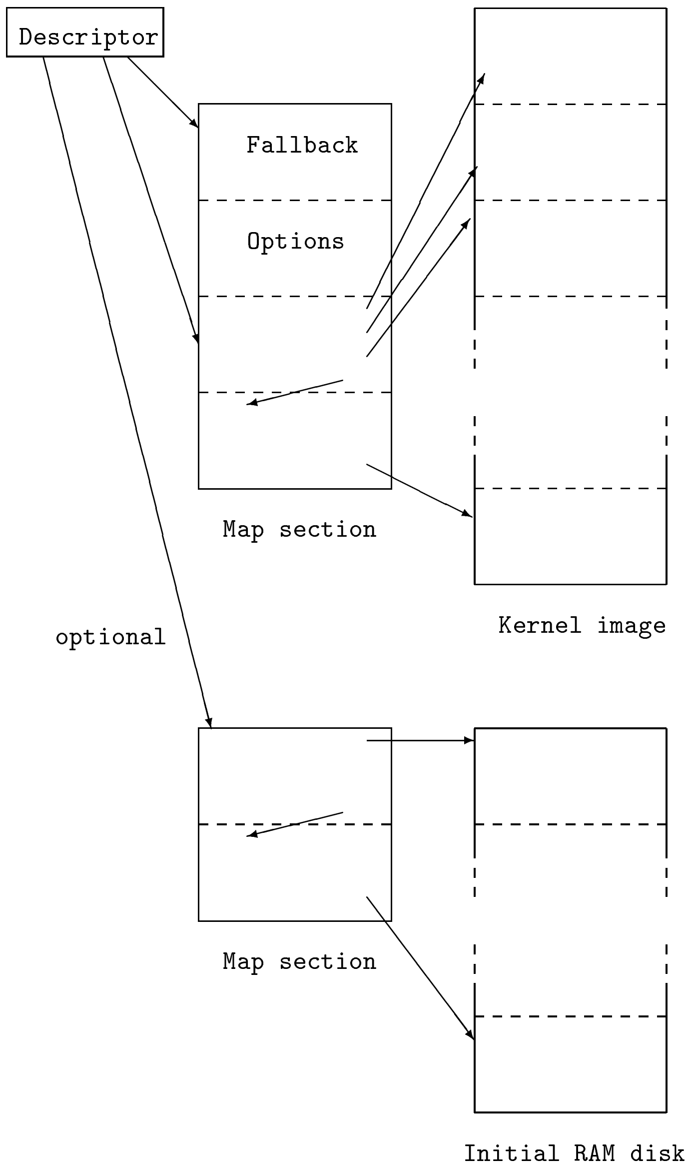


The map file consists of so-called sections and of special data sectors. Each section spans an integral number of disk sectors and contains addresses of sectors of other files.

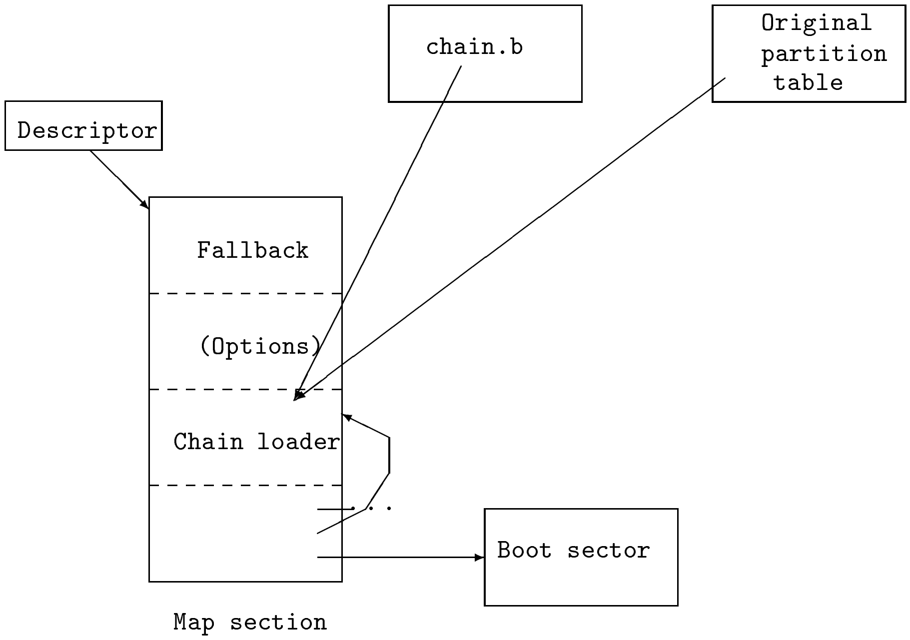
There are three exceptions: 1. If a “hole” is being covered or if the floppy boot sector of an unstripped kernel has been omitted, the address of the zero sector is used. This sector is part of the map file. 2. When booting a different operating system, the first sector is the merged chain loader that has been written to the map file before that section. 3. Each map section describing an image is followed by a sector containing the options line of that image.

The last address slot of each map sector is either unused (if the map section ends in this sector) or contains the address of the next map sector in the section.

The ifive sectors at the beginning of the map file are special: the first sector contains the default command line, the next two sectors contain the boot image descriptor table and the fourth sector is filled with zero bytes. This sector is mapped whenever a file contains a “hole”. The fifth sector contains the keyboard translation table.



A kernel image consists simply of a sequence of sectors being loaded. The map section also contains a sector with a fallback command line and a sector with parameter line options. Optionally, a RAM disk image, specified by a second map section, can be loaded.



When booting another operating system, the chain loader (`chain.b`) is merged with the patched partition table[2] and written into the map file. The map section of this boot image starts after that sector and contains only the address of a dummy floppy boot sector (the zero sector, but its contents are irrelevant), the loader sector and the boot sector of the other operating system. Not that the map section also contains the fallback sector and a (useless) sector for options.

3 Configuration parameters

The boot sector of each kernel contains a set of configuration parameters that have to be available at boot time before the kernel can access file systems. These parameters can be set when the kernel is compiled and later be changed with programs like `rdev`. LILO can supersede the parameters (in memory) at boot time by placing the corresponding items on the parameter line passed to the kernel.

The parameters are stored at the following (decimal) offsets:

497 the size of the setup code in sectors (512 bytes). Older kernels may put a zero at this place.

498-499 is a flag specifying whether the root file system should be mounted read-only (if non-zero) or read-write (if zero).

500-501 the size of the kernel, counted in paragraphs (16 bytes).

502-503 this parameter is currently unused.

504-505 the size of the RAM disk in kilobytes. No RAM disk is created if this parameter is set to zero.

506-507 the text mode the VGA is set to.

- **0xFFFD** the user is asked to specify the VGA mode at boot time.
- **0xFFFE** uses 80x50 ("extended") mode.
- **0xFFFF** uses 80x25 ("normal") mode.

[2] If the partition table is omitted, that area is filled with zero bytes.

Any other value selects the corresponding mode as displayed in the interactive VGA mode selection menu. This is the only option that is set by LILO by patching the boot sector instead of passing it on the parameter line.

508 the minor number of the device that should be mounted as root.

509 the major number of the device that should be mounted as root.

4 Parameter line interface

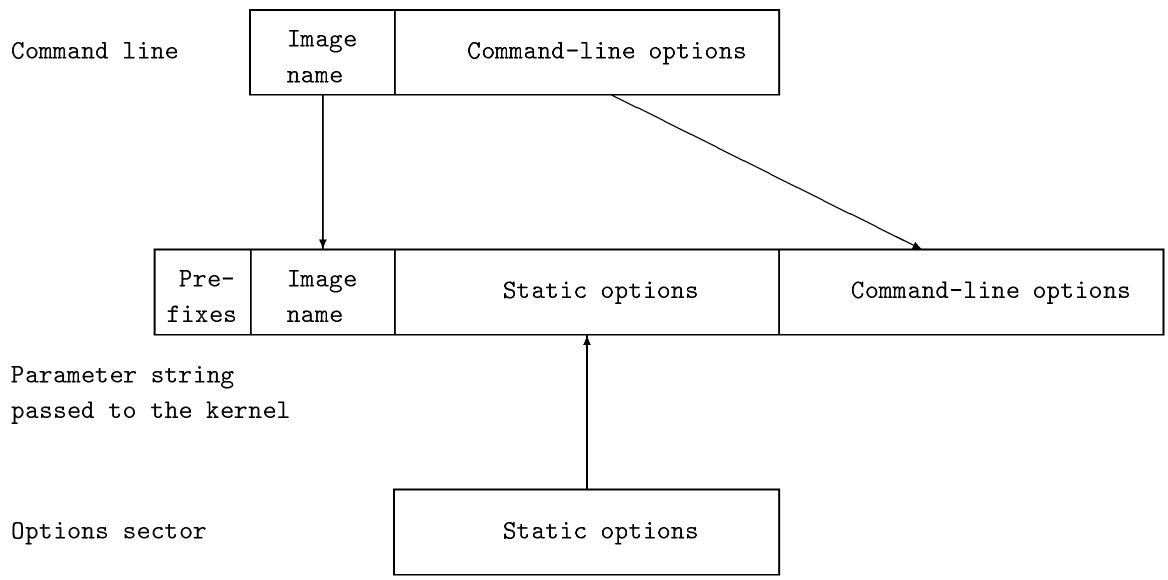
The kernel supports processing of parameters that are provided by the boot loader. The parameter string is a NUL-terminated ASCII string that contains space-separated words or *variable=value* pairs. A description of how they are interpreted can be found in the section of the user's guide labeled "The boot prompt".

The following descriptor has to be set up to pass a parameter string to the kernel:

0x90020 the magic number 0xA33F.

0x90022 the offset of the first byte of the parameter line relative to 0x90000.

The boot loader composes the parameter line from the command line, from the options sector and from some internally generated prefixes (typically `auto` and `BOOT_IMAGE=`), as follows:



Example:
Command line: `vmlinuz root=802`
Options sector: `root=801 ro`

yields `BOOT_IMAGE=vmlinuz root=801 ro root=802`

Because parameter line options can typically be overridden, the first `root` option is ignored by the kernel.

5 External interface

LILO is able to receive its command line from a program that is booted before it. This externally provided command line is only used if the user does not use the normal mechanism to invoke the boot prompt.

The following register contents are expected:

DL contains the value 0xFE.

ES:SI points to the string "LILO". The string must be in upper case and no terminating character is needed. The string must not cross segment boundaries, i.e. **SI** must be below 0xFFFD.

ES:BX points to a NUL-terminated string that is used as the command line. This string has a maximum length of 78 characters (not including the terminating NUL) and must not cross segment boundaries.

There are two values of the externally provided command line that have a special meaning:

- an empty string (**ES:BX** points to a NUL byte) is interpreted as a request to enter the boot prompt and to accept keyboard input.
- a string that consists only of blanks is interpreted as a request to boot the default boot image.

LILO can also obtain the default command line from the map file. It is only used if no externally provided command line is available.

6 Default command line in map file

The first sector of the map file is reserved for a default command line. Unless the user invokes the boot prompt by pressing a shift key or unless an externally provided command line is present, the command line in the map file is interpreted as if it had been typed on the keyboard.

The first two bytes of the first sector of the map file have to contain the magic number `DC_MAGIC` (0xF4F2) in little-endian byte order. They are followed by a NUL-terminated string with a maximum length of 510 bytes, including the NUL. Note that the boot loader limits command lines to 78 characters after removing duplicate spaces.

The command line is disabled by either clobbering the magic number or by using an empty string (i.e. only a NUL byte) as the command line.